Experimental Analysis of P2P Shared-Tree Multicast on MANETs: the Case of Scribe

Franca Delmastro, Andrea Passarella, Marco Conti
Pervasive Computing & Networking Laboratory (PerLab)
IIT Institute, CNR, Pisa, Italy
{firstname.lastname}@iit.cnr.it

***Abstract*— Group-Communication (GC) applications are a promising class for real-world MANETs. To understand their practical viability, we have implemented a full-stack prototype in a real MANET testbed, running an extensive set of experiments on it. In this paper we analyse the performance of a P2P shared-tree multicast protocol to support GC applications. Specifically, we have used Scribe, a popular and efficient solution meant for the legacy Internet. Unfortunately, our results show that such solution is not suitable for MANETs. We identify the Scribe *structured* multicast approach as the main cause of inefficiency, and discuss why *structure-less* solutions may be preferable.**

I. Introduction

Research on 802.11 multi-hop ad hoc networks (MANETs) dates back to almost ten years ago. While the research community has been very active in this field, MANET technologies have not yet had significant impact on the daily life. We believe that a reason is the fact that researchers have privileged a simulation and/or theoretical-driven approach to analyse MANETs' behavior. Simulation and theoretical analysis are very helpful tools since they allow to manage large scales, control parameters' variation, etc. However, propagation over wireless medium is so complicated to precisely model, that experimenting on real test-beds is a must. Just relying on simulation and theoretical models might lead to conclusions that do not match real measures [1], [2]. Furthermore, little effort has been devoted to analyse MANETs behavior with respect to realistic applications. Much effort has been spent on evaluating the behavior of single protocols (mostly at the routing layer) using CBR- or FTP-like applications. Just a few works (e.g., [3]) try to envision application scenarios in which MANETs could be an enabling factor for applications, rather than a hostile networking environment to cope with.

In this paper we investigate the networking requirements of Group-Communication (GC) applications in MANET environments. Specifically, we focus on a distributed whiteboard (WB) running on MANET nodes. WB allows users to share content within members of a community, thus representing an interesting example of MANET-oriented application (see Section II-A). To support WB, we use a P2P middleware platform made up of a structured overlay network, and an application-level multicast protocol. This choice might not look intuitive. In fact, also standard routing-level multicast allows group communications, but P2P systems provide several other valuable services for GC applications. In particular, they provide a decentralised, self-organising and self-healing environment, along with features like subject-based routing, distributed data storage/retrieval, and load balancing. Such features, originally devised for P2P overlay networks in the legacy Internet, are also well suited for a decentralised and dynamic environment like a MANET, even though some optimizations are needed to improve their performance [4], [5], [6].

In this paper we mainly focus on the multicast protocol performance and WB usability. To this aim, we have implemented a full-stack prototype (including all layers from the physical up to the application) in a real-MANET testbed, running an extensive set of experiments. In our prototype we used Pastry [7] and Scribe [8] to implement the overlay network and the multicast protocol, respectively (see Section II). This choice was motivated by results reported in [9] that show how this platform outperforms other ones (e.g., CAN) providing equivalent services.

Experimental results, with special emphasis on the performance of Scribe, are discussed in Sections IV and VI. Specifically, we focus on the QoS provided to WB users in terms of average delay and packet loss. Purposely, we report only results gathered from static MANETs. Taking also into account users mobility would have added further dimensions to the parameters space, making results quite difficult to understand. Outcomes of these experiments highlight severe limitations of Scribe in supporting GC applications over MANETs. Even though refined software releases might improve the user QoS (Section III), there are intrinsic Scribe features

that hinder from using it in MANETs. They mainly stem from the design choice of concentrating all the application-level traffic on one single node (the Scribe Root Node) before delivering it to final destinations.

Our results highlight two main bottlenecks resulting from this choice. On one hand, the root node is very likely to be overloaded by the traffic generated by non-root nodes. In this case, application-level messages are discarded at source nodes, or suffer very high delays. In Section IV we show that in such cases GC applications can be reasonably used just with light traffic load (i.e., below 10 packets per second generated by each user). On the other hand, we have also found that, as soon as few nodes are more than 2-hop away from root, the whole system performance drastically decreases.

Finally, in Section VI we highlight that the structured multicast approach used by Scribe is one of the main reasons of these inefficiencies. Specifically, structured multicast tends to concentrate the application load on few nodes and links that may become easily overloaded. Therefore, we highlight that structure-less multicast approaches can avoid such concentration, representing an interesting possibility to support GC applications in MANETs.

A. Related work

Experiment-based research on wireless networks, and particularly on MANET, is gaining momentum in the last few years [10], [11]. Having controllable, reproducible and reasonable-size wireless testbeds is not trivial. Thus, several research efforts are focusing on how to design and implement testbeds that the whole community can exploit [12], [13], [14], [15]. One of the main issues of simulation and theoretical analysis is the accuracy of wireless channel models. Therefore, a number of papers analyse wireless channel features aiming at providing realistic models (see, for example, [1] and references herein). Other research efforts target the experimental evaluation of routing [16], [17] or transport [10], [11] protocols on MANETs.

This paper complements our previous works on experimental analysis of middleware platforms for MANETs. The work in [4] and [6] focuses on issues related to structured overlay networks. Specifically, these papers compare Pastry and CrossROAD [18], an optimised P2P overlay network for MANETs. They show how cross-layer interactions with a proactive routing protocol can be a key to implement structured overlay networks over MANETs in a very efficient way. In [19] we started analysing the performance of GC applications on MANETs, showing the advantages of using CrossROAD to efficiently support them. Instead of focusing exclusively on the overlay network, this paper analyses how a legacy P2P multicast protocol (i.e., Scribe) works in this scenario.

II. Experiments setup

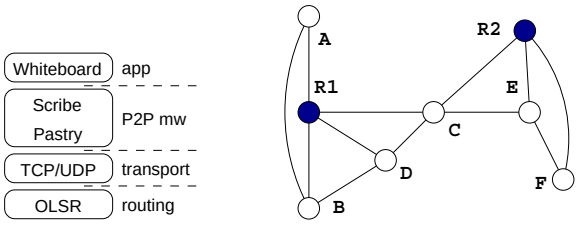
A. Application and Protocol Stack

One of our targets is to envision realistic MANET applications and understand how they should be developed in practice. From this standpoint, GC applications are quite interesting. They fit well the overall features of MANETs since they are distributed, self-organising, and decentralised in nature. As a simple - yet significant - example, we developed a Whiteboard application (WB), which implements a distributed whiteboard among MANET users. WB usage is very intuitive. Users run a WB instance on their own devices, select a topic they want to join, and start drawing on the canvas. Drawings are distributed to all devices subscribed to that topic, and rendered on each canvas. WB is just an example of a broader range of applications, including distributed messaging, distributed gaming, etc. We believe that this kind of applications can be valuable to MANET users, and they can thus contribute to bring MANET technologies into everyday life.

WB has been developed on top of the network protocol stack shown in Figure 1(a). Specifically, WB runs on top of a P2P middleware layer made up of a structured overlay network (Pastry [7]), and an application-level multicast protocol (Scribe [8]). WB maps each interest group (i.e., each topic) to a multicast tree, and exploits the multicast protocol services to deliver information to group members. At the routing layer we used OLSR [20] since, in previous experiments on our testbed ([5], [4], [19]), it has shown to outperform reactive routing protocols (namely, AODV [21]) without generating significantly higher overhead. Previous results also showed that Pastry is characterized by some inefficiencies in ad hoc environments. To deal with them, an optimised solution (CrossROAD [18]), which exploits main features of a cross-layer architecture [22], has been defined.

In this paper we do not take into account any cross-layer optimization. Before designing a completely new, optimized, multicast system, we want to understand whether a legacy solution could be suitable for GC applications on MANETs, and in which directions it should be improved, if the case. Since Pastry and Scribe [8] have shown to outperform other similar solutions in the legacy Internet [9], we start from their evaluation in a real testbed[1].

[1]Actually, we used the free implementation of the Rice University, included in the FreePastry package [23].



(a) protocol stack (b) network topology

Fig. 1. Protocol stack and network topology of our testbed.

Pastry implements an overlay network in the form of a Distributed Hash Table (DHT). Each node gets a logical address within a circular address space as the hashed value of its IP address. A key is attached to each message to be sent on the overlay. The final destination of the message is the node with the closest address (in the circular space) to the hashed key. The onus of defining a key for each message lies with the layer running on top of Pastry (Scribe in this case).

Scribe exploits the DHT to build shared trees among groups of nodes. Each tree is identified by a topic. Scribe defines a Root Node, as the node in the overlay whose address is the closest one to the hashed topic. In Figures 2 and 3 the root is node C. Each node willing to join the tree sends a subscribe message specifying the topic as the key. If the local node does not directly know the root of the topic, the message is forwarded using a multi-hop route at the middleware level, which might require a higher number of hops at the routing level. An intermediate node receiving such message either subscribes itself to the same tree by sending its own subscribe message towards the root (e.g., node B after step 1 in Figure 2), or discards the message if it is already a member of the tree (e.g., node B in step 3 in Figure 2). In both cases, it enrolls the node from which it received the message as a child. Messages to be delivered over the tree are first sent towards the root of the topic (step 1 in Figure 3), and subsequently delivered by each parent to its children (steps 2 and 3 in Figure 3). Note that Pastry uses TCP connections to carry these application messages.

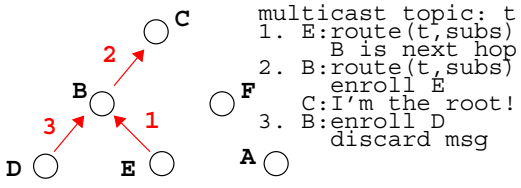


Fig. 2. Scribe building the tree

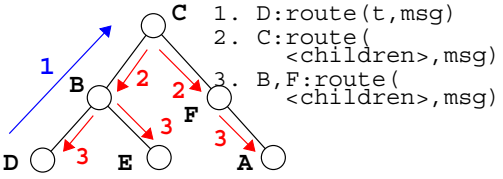


Fig. 3. Scribe delivering data

B. Network topology

Figure 1(b) shows the network topology we used. In each trial nodes A to E ran the whole protocol stack, including the WB application, while nodes R1 and R2 just worked as routers. We think this is a reasonable scenario for standard, stand-alone MANETs. Specifically, it lies within the ad hoc horizon defined in [2], i.e., up to 10-20 nodes, and up to 2-3 hops. Theoretical [24] and experimental [2] results show that MANETs beyond this horizon are unable to deliver reasonable throughput to users, and thus they are not likely to be really deployed.

C. Experiments definition and performance indices

In our experiments all nodes were IBM ThinkPad R50 laptops with integrated 802.11b wireless card (Intel PRO-Wireless 2200). The OS was `linux-2.6.12.3`, loading the `ipw2200` driver for the network card. The experiment software can be downloaded from `http://bruno1.iit.cnr.it/scribe_exp_sw/`.

In all the experiments nodes A to E ran the WB application. Specifically, a single tree joined together these nodes, that were thus members of the same group. Users interactions were simulated by the application alternating between active and idle phases. Specifically, in each active phase the application generated traffic on the network corresponding to strokes drawn on the WB. Both the number of strokes drawn during an active phase, and the duration of an idle phase were exponentially distributed. Such a traffic profile is bursty, representing the typical behavior of a user that sends content to be shared with the group, and then "idles", looking at the data generated by other users.

We ran experiments by varying both the average idle-phase duration, and the average number of strokes per active phase. Each trial was composed by 100 active/idle cycles, and we took care that each node running WB generated at least 100 messages[2]. To make trials start at the same time at different nodes, we synchronised the nodes before each trial, and scheduled the trial start at the same time on each node.

[2]A distinct message was sent for each stroke. The size of each message was 1448 bytes.

In the following, each trial is identified by the application-load index, measured as the number of Packets Per Second (pps) generated by each user. This index is defined as the ratio between the average number of strokes generated in a cycle, and the average duration of an active/idle cycle. We have found that this simple index is sufficient to correctly identify usage cases for our scenario.

The main performance indices presented in the following are the packet loss and the average delay experienced by each node during the experiment. The packet loss is defined as $1 - \frac{R_i}{\sum_{j=1}^{N} S_j}$, where R_i is the number of messages received by node i, and S_j is the number of messages transmitted by the j-th node. The average delay experienced by node i is defined as $\sum_{j=1}^{R_i} d_{ij}/R_i$ where d_{ij} is the delay experienced by node i in receiving packet j. We have also defined a usability threshold for the application, indicating reference values for both delays and packet loss. Beyond these thresholds the application performance is not compliant with users expectations. To have reasonable values, in our case we assume 10s delay and 15% packet loss as thresholds. Of course they closely depend on the specific application requirements. We replicated each configuration several times, obtaining quite variable results due to the characteristic variability of the wireless medium. In this paper we show the best results measured in each configuration.

III. Evolution of software releases

In [19] we reported a first set of experiments run with the FreePastry 1.3 release. We found that the Pastry/Scribe platform was practically unable to support even light application loads. Afterwards, we used one of the latest versions of FreePastry (1.4.1) to compare system performance and highligth improvements introduced by software releases. Specifically, Figures 4 and 5 show respectively the performance we measured in terms of average delay and packet loss for both software releases. Each figure shows the performance measured in the best case (that, clearly, is achieved by the Scribe Root Node, since all WB messages have to be firstly sent to it), and the average performance achieved by non-root nodes. Let us firstly focus on "Pastry 1.3" curves. The Scribe Root Node experiences a reasonable QoS for all application loads (i.e., 0.2 pps, 0.5 pps, and 0.8 pps). Specifically, the root-node average delay is always below 5s, and the packet loss below 15%. But other nodes obtain a largely unsatisfactory service. We can identify a critical point for an application load between 0.2 and 0.5 pps. Even though the average delay at 0.5 pps is about 2.3s, the packet loss increases to 32.86%. Note that beyond this critical point the system becomes pretty unstable and quite unpredictable. When the application load increases to 0.8 pps, the packet loss slightly decreases to 26.50%, while the average delay increases to 13.18s. Overall, the system is reasonably usable only for very light application loads, and deploying applications like WB on this platform becomes quite questionable.

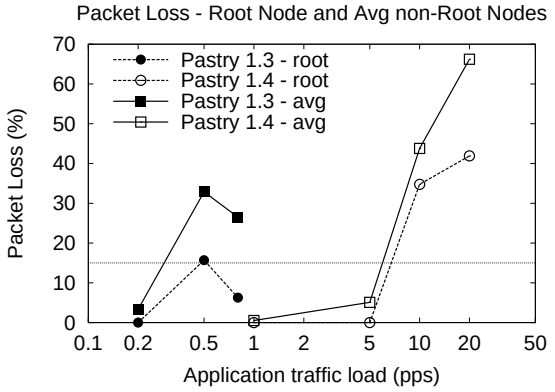


Fig. 4. Packet loss.

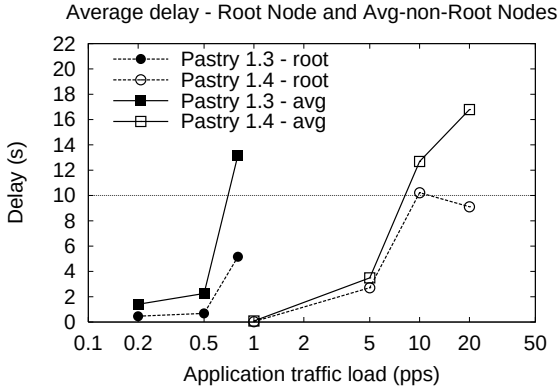


Fig. 5. Average delay.

Let us now focus on "Pastry 1.4" curves. FreePastry 1.4.1 has introduced major modifications to the overlay building and maintenance procedures, drastically reducing the overhead and improving the overlay stability [23]. Thus, it was interesting to explore if this new release improves the performance in our scenario.

By looking at Figures 4 and 5 the performance improvement is evident. The critical point moves by more than one order of magnitude, lying now between 5 and 10 pps. Indeed, at 5 pps also non-root nodes experience reasonable QoS, since the average delay is about 3.5s, and the packet loss is 5%. On the other hand, at 10 pps and beyond the application becomes hardy usable at any node. Note that, even though the average delay at root node would be almost acceptable also at 10 and 20pps

(i.e., it is below the usability threshold), the packet loss increases to 35% and 42%, respectively.

Note also that between 10pps and 20pps the delay curve relative to the root node flattens. This is actually a side effect of the higher packet loss experienced at 20pps. In detail, in the topology of our experiments, non-root nodes are either 1 or 2 hops away from root. In the next section we show that even at 20pps 1-hop-away nodes are able to send to root almost all the traffic locally generated. The additional packet loss experienced by root at 20pps is thus due to less messages received from 2-hop-away nodes. In other words, out of the whole bunch of messages received by root, the fraction of messages received by 1-hop-away nodes *increases* when the traffic load shifts from 10 to 20pps. Since messages from 1-hop-away nodes experience significant lower delay than messages from 2-hop-away nodes, the fraction of messages experiencing low delay increases too. This reduces the average delay measured at root, resulting in the flat shape of the curve. The same phenomenon does not apply to non-root nodes. Recall that messages have to reach the root before being delivered to other nodes. The delay experienced by non-root nodes in receiving messages *from root* increases between 10 and 20 pps. Thus, even though in both cases messages experience – on average – the same delay between the originating node and root, in the 20 pps case they undergo higher delay along the path between root and the final destination.

To summarise, the above analysis shows a steep improvement in the user-perceived QoS when moving from a FreePastry release to a more advanced one. However, a critical point still exists, beyond which it practically makes no sense to use GC applications like WB on this platform. At this point it is important to understand whether this critical point is going to eventually disappear thanks to future software releases, or it is intrinsic to the Pastry/Scribe design. In the following sections we analyse the results achieved with FreePastry 1.4.1 more in depth, and show that, independently of software refinements, the design of Scribe includes features not suitable for the MANET environment.

IV. User QoS: a rather optimistic setup

In this set of experiments we have placed the root node at the center of the topology to minimise the average hop distance to any other node. Since it is well-known that TCP performance drastically worsen as the hop distance increases ([2], [5]), this represents an optimistic setup. To have a clearer picture of the system behavior, we now focus on the average delay and packet loss experienced by each single node. Specifically, curves in Figure 4 and 5 show the average performance experienced by non-root nodes, and thus provide indications about the average QoS a user may expect. In this section we analyse the performance of nodes at 1 and 2 hops from root separately. Together with the "root-curves" in Figures 4 and 5 this provides a precise view of the expected QoS with respect to the nodes' position in our topology.

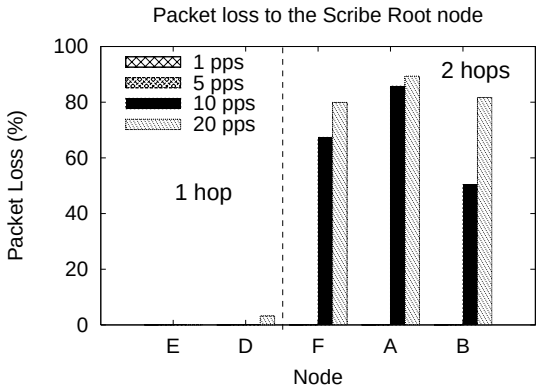


Fig. 6. Packet loss towards root

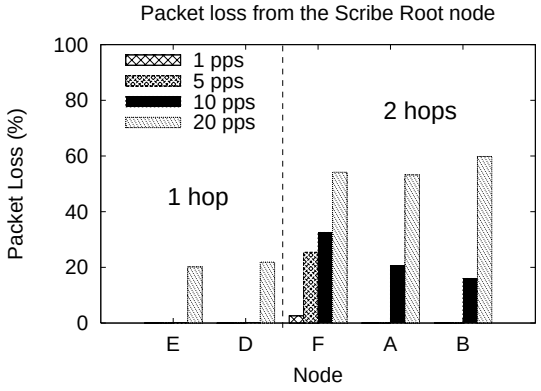


Fig. 7. Packet loss from root

Figures 6 and 7 show the packet loss experienced by each single node *towards* and *from* root, respectively. The packet loss of the i-th node *towards* root is defined as the ratio between the number of messages generated at the i-th node and not received by root, and the number of messages generated at the i-th node. The packet loss of the i-th node *from* root is defined as the ratio between the number of messages not received by the i-th node out of the total number of messages sent by root, and the total number of messages sent by root. Note that, due to the Scribe architecture, root not only sends messages locally generated, but also messages that it receives from all the other nodes. In addition, the i-th node experiences packet loss 0 only if *i)* the packet loss of *all* nodes *towards* root is 0, and *ii)* the packet loss of the i-th node *from* root is

0.

On one hand, Figures 6 and 7 confirm that, as far as the packet loss, the critical point for WB usability lies between 5 and 10pps. Indeed, below 5pps the packet loss *towards* root is 0 at all nodes. This means that all nodes are able to send messages locally generated to root. Apart from node F, also the packet loss *from* root is 0 at all nodes. Thus, the overall packet loss experienced by all nodes (not shown here for the sake of space) is 0 at 5pps and below, making WB usable.

On the other hand, Figures 6 and 7 show a drastic difference between nodes at 1 hop and 2 hops distance from root. Even though the magnitude of this difference is quite surprising, a steep perfomance decrease in the case of multi-hop connections was expected (see, for example, [2], [5]). Specifically, Figure 6 shows that, even at 10 pps and 20 pps, 1-hop-away nodes are able to send all their messages to root. Instead, 2-hop-away nodes see their outgoing traffic cut by 50% to 89%. Figure 7 shows a similar trend, in the sense that 2-hop-away nodes experience a far higher packet loss than 1-hop-away nodes at 10 and 20 pps. However, there is a difference worth to be noted. While for 2-hop-away nodes the packet loss is higher in the direction *towards* root, for 1-hop-away nodes the packet loss is higher in the direction *from* root. At a higher level, one could note that as the traffic load increases, Scribe cuts it because the root node becomes overloaded. In our configuration, while 1-hop-away nodes suffer only in the direction *from* root, 2-hop-away nodes mainly suffer in the direction *towards* root. Actually, we have found configurations in which for both cases (i.e., 1 and 2 hops) the main traffic cut occurs in the direction *from* root (results are not shown here due to lack of space). Understading the reason of this behavior is not trivial, thus we are currently analysing the system even more deeply. However, it should be noted that, as far as the application-level QoS, the precise direction along which the main traffic cut occurs is not that important.

Figures 6 and 7 finally show that the presence of 2-hop-away nodes makes the application unusable also for 1-hop-away nodes. For example, let us focus on the 10 pps case. Nodes 1-hop away from root measure 0 packet loss on both directions. However, they are unable to receive most of the messages generated at nodes 2-hops away from root, because those nodes suffer very high packet loss *towards* root. This highlights that, since all messages have to be firstly sent to the root, a poor connection between a particular node and root makes *all* other nodes unable to receive the messages generated by this node.

Analysing the delay figures allows us to highlight a further feature of the system. Specifically, Table I shows the average delay and the main percentiles depending on the application-traffic load, and on the hop-distance from root. By looking at the average delays only, one could conclude that the application is usable even at 10pps by nodes at most 1-hop away from root. However, defining the usability threshold with respect to the 90th percentile instead of the average value, the critical point shifts below 5pps (for all nodes).

	avg/percentiles	root	1 hop	2 hops
1pps	avg	0.032	0.055	0.100
	90	0.070	0.115	0.221
	95	0.126	0.159	0.316
	99	0.282	0.321	0.604
5pps	avg	2.710	2.900	3.924
	90	11.00	11.29	13.15
	95	13.74	13.95	16.62
	99	23.19	23.28	25.71
10pps	avg	10.22	10.57	14.52
	90	34.06	34.79	43.11
	95	65.06	65.11	71.36
	99	101.4	101.4	101.8
20pps	avg	9.11	13.12	21.15
	90	23.09	31.11	88.03
	95	86.30	89.30	115.1
	99	145.4	145.7	146.4

TABLE I
DELAYS DEPENDING ON THE HOP-DISTANCE FROM ROOT (SECONDS).

Table I also shows a drastic difference between 1pps and the other traffic loads. Indeed, at 1pps WB performance are completely satisfactory, as the 99th percentile for 2-hop-away nodes is about 600ms. At higher traffic loads there is a significant difference between the average delays and the 90th percentiles. This suggests that the delay distributions have a long tail. This is confirmed by looking at Figure 8, which plots the CCDF of delays measured at root node. Clearly, for each traffic load, the CCDF at root is a lower bound of CCDFs at any other node. Figure 8 shows that CCDFs for application loads of 5pps and beyond can be lower bounded by a Pareto distribution with parameter 0.25. Specifically, they show a long-tail pattern in the range from 5ms to 10s. At the application level, this means that, even though the average delay can lay below the usability threshold, delay values are highly variable, and thus no strict QoS guarantees can be granted.

From results presented in this section we can conclude that the centralised approach of Scribe generally hinders the use of GC applications over MANETs. Specifically, we have highlighted two main inefficiencies:

- even few nodes poorly connected to root prevent *all* nodes from using GC application properly;

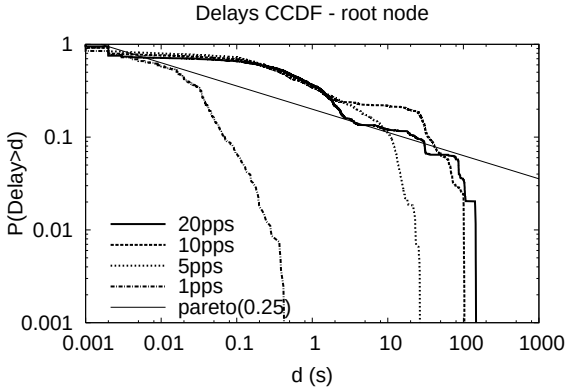


Fig. 8. Delay CCDF measured at root

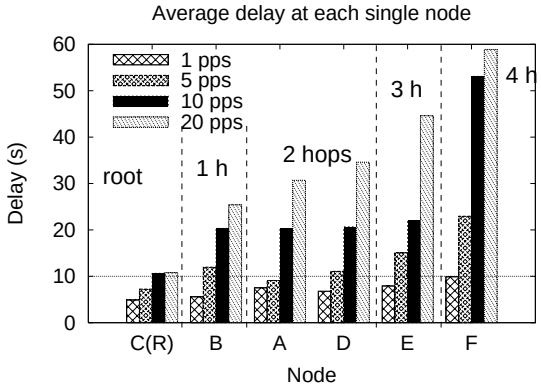


Fig. 9. Average delay

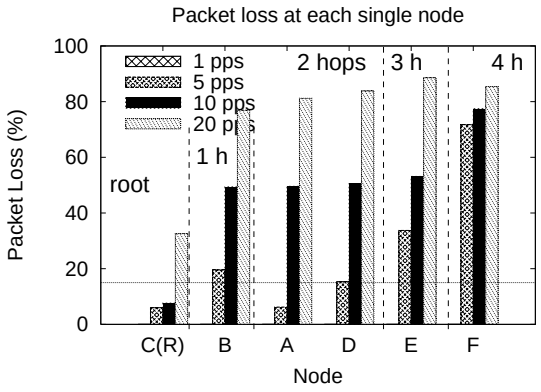


Fig. 10. Packet Loss

- the root node very likely experiences a long-tailed delay distribution. In these cases *any* other node experiences the same pattern in the delay distribution.

Both these drawbacks are intrinsic to the Scribe design, and cannot be removed by refined software releases. Note also that our experiments were run with quite powerful laptops. Performance could thus be even worse in case of less powerful devices, such as PDAs, that are natural candidates to be used in MANET environments.

V. System performance varying the Scribe Root Node location

In the experiments presented so far, the Scribe Root Node was always at the center of the network topology and all other nodes were 2 hops away at most. However, in the general case no assumptions about the root location can be done. Therefore, we ran a further set of experiments by placing the root node at one edge of the network. This might sound like a pessimistic configuration, but it should be noted that having the root node at one edge of the network is more likely than having it at the center of the topology. In detail, with respect to the topology in Figure 1(b), we swapped the positions of nodes C (root node) and A. This setup also allows us to better analyse the impact of longer paths on Scribe, since we now have TCP connections spanning 1 to 4 hops.

Figures 9 and 10 show the performance measured at each single node in terms of average delay and packet loss respectively. They confirm that the system now is usable just at lighter application loads. Even at 1 pps, while the packet loss is 0 at all nodes, the average delay ranges between 5 and 10s. Thus, even at 1 pps, the system is usable just for applications with loose delay constraints. At a load of 5 pps the system is completely unusable for nodes 3-hop away from root and beyond. At higher loads, the performance might be too low even for the root node.

VI. Discussion and future work

Scribe was designed having in mind a resource-rich environment, such as the legacy wired Internet, and it has shown to perform very well in it [9]. However, when used over MANETs, the centralised approach, in which a root node is in charge of delivering all application messages, leads to very low performance in terms of packet loss and delay.

More in general, we believe that one of the main reasons of these low performance is the use of a structured multicast solution. Apart from requiring significant overhead in terms of management traffic, these solutions tend to concentrate the costs of the application (in terms of network/computation resources) on a few nodes and links. If these nodes/links are underprovisioned, or happen to be placed in adverse locations, the whole system may implode, making it unable to support the application. Structured solutions are a good choice in large-scale systems designed for the legacy Internet (like Scribe is). Indeed, in this case the network and computational resources are not a big issue, and structured solutions

allow the system to scale up to thousands of nodes. However, growing up to such a scale is not reasonable for MANETs. Theoretical results and practical experiences (e.g., [24], [2]) show that the most reasonable scale for MANETs based on 802.11 technologies is up to 10-20 nodes, and 2-3 hops. Furthermore, concentrating the load on a few resources and managing the multicast structure may become serious problems in MANETs.

Based on these remarks, we believe that *structure-less* multicast be a more reasonable choice in this environment. For example, approaches like Differential Destination Multicast (DDM, [25]) and Route Driven Gossip (RDG, [26]) seem to be very interesting solutions. Each member of a multicast group knows the other members of the same group (actually, RDG also works with partial knowledge). When a message is locally generated, it is sent to the group members by using the underlying routing protocol, without requiring any multicast structure[3]. These approaches spread the load more evenly over the nodes and links of the network, avoid concentration on a few nodes/links, and typically work remarkably well in small to medium size networks. However, they require costly mechanisms to collect and maintain the (partial) list of group members at each node. Typically, nodes have to periodically flood the network to announce their presence, or to check for other nodes' liveness. Therefore, we are currently designing an improved version of Scribe, that retains the structure-less features of DDM and RDG, but avoids such costly mechanisms by exploiting a cross-layer approach.

REFERENCES

[1] G. Anastasi, E. Borgia, M. Conti, E. Gregori, and A. Passarella, "Understanding the real behavior of mote and 802.11 ad hoc networks: an experimental approach," *Pervasive and Mobile Computing*, vol. 1, no. 2, pp. 237–256, July 2005.

[2] P. Gunningberg, H. Lundgren, E. Nordstroem, and C. Tschudin, "Lessons from experimental manet research," *Ad Hoc Networks Journal*, vol. 3, no. 2, pp. 221–233, Mar. 2005.

[3] E. Huang, W. Hu, J. Crowcroft, and I. Wassell, "Towards commercial mobile ad hoc network applications: a radio dispatch system," in *Proc. of ACM MobiHoc*, Urbana, IL, May 2005.

[4] E. Borgia, M. Conti, F. Delmastro, and E. Gregori, "Experimental comparison of Routing and Middleware solutions for Mobile Ad Hoc Networks: Legacy vs Cross-Layer approach," in *Proc. of the ACM SIGCOMM E-WIND Workshop*, Philadelphia, Aug. 2005.

[5] E. Borgia, M. Conti, F. Delmastro, and L. Pelusi, "Lessons from an Ad-Hoc Network Test-Bed: Middleware and Routing Issues," in *Ad Hoc & Sensor Wireless Networks, An International Journal*, Vol.1, Numbres 1-2, 2005.

[6] E. Borgia, M.Conti, F. Delmastro, and A. Passarella, "MANET perspective: current and forthcoming perspective," in *Proc. of IST Mobile and Wireless Communication Summit 2006*, Myconos, June 2006.

[7] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object localtion and routing for large scale peer-to-peer systems," *LNCS*, vol. 2218, pp. 329–350, 2001.

[8] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron, "Scribe: A large-scale and decentralised application-level multicast infrastructure," *IEEE JSAC*, vol. 20, no. 8, Oct. 2002.

[9] M. Castro, M. B. Jones, A.-M. Kermarrec, A. Rowstron, M. Theimer, H. Wang, and A. Wolman, "An evaluation of scalable application-level multicast built using peer-to-peer overlays," in *Proc. of INFOCOM*, San Francisco, CA, Apr. 2003.

[10] (2005, July) IEEE ICPS REALMAN Workshop. Santorini, Greece. [Online]. Available: http://www.cl.cam.ac.uk/realman/05/

[11] (2005, Aug.) ACM SIGCOMM E-WIND Workshop. Philadelphia, PA. [Online]. Available: http://www-ece.rice.edu/E-WIND/

[12] Department of Computer Systems at Uppsala (Sweden). Ape: Ad hoc protocol evaluation testbed. [Online]. Available: http://apetestbed.sourceforge.net/

[13] N. Vaidya, J. Bernhard, V. Veeravalli, P. Kumar, and R. Iyer, "Illinois wireless wind tunnel: A testbed for experimental evaluation of wireless networks," in *Proc. of the ACM SIGCOMM E-WIND Workshop*, Philadelphia, PA, Aug. 2005.

[14] K. Ramachandran, S. Kaul, S. Mathur, M. Gruteser, and I. Seskar, "Towards large-scale mobile network emulation through spatial switching on a wireless grid," in *Proc. of the ACM SIGCOMM E-WIND Workshop*, Philadelphia, PA, Aug. 2005.

[15] P. De, A. Raniwala, S. Sharma, and T. Chiueh, "Mint: A miniaturized network testbed for mobile wireless research," in *Proc. of INFOCOM 2005*, Miami, FL, Mar. 2005.

[16] E. Borgia, "Experimental Evaluation of Ad Hoc Routing Protocols," in *Proc. of IEEE PerCom PWN Workshop*, Kauai Island, Hawaii, Mar. 2005.

[17] R. S. Gray, D. Kotz, C. Newport, N. Dubrovsky, A. Fiske, J. Liu, C. Masone, S. McGrath, and Y. Yuan, "Outdoor Experimental Comparison of Four Ad Hoc Routing Algorithms," in *Proc. of MSWiM 2004*, Venice, Italy, October 2004.

[18] F. Delmastro, "From Pastry to CrossROAD: Cross-layer Ring Overlay for AD hoc networks," in *Proc. of IEEE PerCom MP2P Workshop*, Kauai Island, Hawaii, Mar. 2005.

[19] F. Delmastro and A. Passarella, "An experimental study of p2p group-communication applications in real-world manets," in *Proc. of IEEE ICPS REALMAN 2005 Workshop*, Santorini, Greece, July 2005.

[20] A. Tonnesen. OLSR: Optimized link state routing protocol. Institute for informatics at the University of Oslo (Norway). [Online]. Available: http://www.olsr.org

[21] AODV: Ad hoc on demand distnce vector routing. Dept. of Information technology at Uppsala University (Sweden). [Online]. Available: http://user.it.uu.se/ henrikl/aodv/

[22] M. Conti, G. Maselli, G. Turi, and S. Giordano, "Cross layering in mobile ad hoc network design," *IEEE Computer*, Feb. 2004.

[23] FreePastry, Rice University. [Online]. Available: http://freepastry.rice.edu

[24] P. Gupta and P. Kumar, "The capacity of wireless networks," *IEEE Transactions on Information Theory*, vol. 46, no. 2, pp. 388–404, Mar. 2000.

[25] L. Ji and M. Corson, "Explicit multicasting for mobile ad hoc networks," *Mobile Networks and Applications*, vol. 8, pp. 525–549, 2003.

[26] J. Luo, P. Eugster, and J.-P. Hubaux, "Probabilistic reliable multicast in ad hoc networks," *Ad Hoc Networks*, vol. 2, pp. 369–386, 2004.

[3]Mechanisms are included to send just once a message addressed to destinations sharing the same next-hop from a sending node.